ACES JOURNAL, VOL. 13, NO. 2, JULY 1998, SI: CEM & HPC

Implementation and Application of a FD-TD Simulation Tool for the Analysis of Complex 3D Structures

Kevin Thomas
Cray Research
655 Lone Oak Road, Eagan, MN 55121
phone 612-683-3624, fax 612-683-3099
kjt@cray.com

Gary Haussmann and Melinda Piket-May
Department of Electrical and Computer Engineering
Campus Box 425, University of Colorado, Boulder,CO 80309
phone 303-492-8719, fax 303-492-5323
Gary.Haussmann@Colorado.edu
mjp@boulder.colorado.edu

Roger J. Gravrok
Sequent Computer Systems
Bldg. D2, Ste. 219 Mailbox 82
800 Wisconsin Street
Eau Claire, WI 54703-3611
phone 715-830-7321, fax 715-833-2027
rjg@sequent.com

Abstract

This paper presents information about the development of an electromagnetic analysis tool "LC" which integrates the Finite-Difference Time-Domain (FD-TD) method with an interactive Graphical User Interface (GUI). The paper will discuss the program implementation and design; many issues in the implementation have surfaced, concerning the problem of producing a graphical model editor/simulator that runs efficiently on various hardware systems. The paper will also explore how the solver's capabilities aid design engineers when investigating and solving packaging and interconnect design issues as well as the program's application to engineering problems.

1 Introduction

At the level of a chip, or a device comprised of several chips with interconnects, the electromagnetic environment is very complex and not necessarily well understood theoretically, numerically, or experimentally. For instance, coupling between vias can distort signals, mismatches between vias and signal lines can lead to ground-bounce, holes in ground planes may result in increased coupling effects between board layers, and metal traces are likely to have reactive impedance components that can degrade system performance at higher clock speeds. The simulation tool discussed in this paper was designed to aid in understanding these effects, as well as to discover means to mitigate and/or overcome these effects. Also, this paper will discuss the development of general methods used to extract relevant information from the simulation.

2 Background

We have chosen to use the Finite-Difference Time-Domain (FD-TD) method as our Maxwell's Equations solver [1]. The FD-TD method is a very powerful computational tool: it gives time-domain data, useful for transient analysis, and can yield frequency-domain data via Fourier Transforms. Taken together, all of these capabilities have made FD-TD an attractive and robust method for solving a number of electromagnetic interaction problems. Previously a major limitation to the full scale industrial deployment of FD-TD tools has been the absence of a versatile GUI to generate models and postprocess the vast amount of data that the FD-TD method can generate.

This paper will discuss an EM analysis development tool called "LC". It consists of a GUI interface with the ability to auto-mesh a graphically defined three-dimensional user geometry. A number of different excitations are possible, and the outputs can yield not only field data, but also voltages, currents, impedances, inductances, capacitances and fluxes. Further, both time and frequency-domain data are available on output. LC also allows for 2-D visual plots of the full 3-D problem being simulated during time-stepping. The ability to visualize the three-dimensional electrodynamics of a structure lends a great deal of intuitive insight and understanding to the user. This is also very useful in identifying problem areas with a particular design.

Another very important and relevant feature of the LC tool is an interface to SPICE [2], which will analyze a circuit based on FD-TD field values linked to SPICE at each time step. The SPICE interface also generates output voltages and/or currents that modify the FD-TD field values which are then used during the next FD-TD time-step. The efficient implementation of LC on scalable parallel computers (under development) will allow for the examination of problems that are currently too computationally large or complex, conceivably making inroads into some of the grand challenges facing the electromagnetic engineering community.

Essentially LC is an integrated EM model editor, simulator, and analysis tool. It's composed of 150,000 lines of C and Fortran and is written to run on Unix. The simulator can use multiple processors in parallel to reduce the solution time on shared-memory multiprocessors. A full description can be found on the LC web page *http://www.cray.com/lc/*. While the program source code is not public domain, a demonstration version (distributed in executable form) is available for many types of computers.

The LC tool is so named because it's initial development was driven by the need for an accurate three-dimensional characterization of the inductance (L) and capacitance (C) of complex electronic systems. Specifically, the creation of LC was motivated by the need to develop techniques and a corresponding tool that were both accurate and suitable for the analysis/design of circuits with clock speeds above 100 MHz. Beyond 100 MHz, circuit interconnect performance dictates system performance capabilities. Accurate electrical characterization of complex electronic structures in three dimensions is mandatory for good design. EM modeling challenges such as those associated with printed circuit boards, multichip modules, integrated circuit packages, connectors, and electromagnetic interference are currently being performed. LC may be used to look at signal- transmission characterization and power distribution modeling. The characterization requirements for these problems include an intuitive and visual understanding of the electrodynamics along with a quantitative description of L,C,R,Z,and T_d parameters.

3 Implementation Details

This section describes the FD-TD details involving mesh setup, boundary conditions, parallelization, and optimization/performance testing. The bulk of the FD-TD engine is based on standard FD-TD updates [1], with slight modifications to allow for straightforward programming to communicate to the user interface code. Other key modifications include the ability to choose from numerous boundary conditions and parallelization of the original serial code.

3.1 Meshing

The FD-TD algorithm in LC is based on a three dimensional Cartesian grid. A rectangular zone enclosing the model space is discretized into uniformly-sized cubic cells; thus a mapping is achieved to an *ijk* (integer-indexed) 3-D grid. Each 3-D cell is assigned a single material definition, based on the material at the center of the cell, making it homogeneous. Each cell is broken down into its six faces, and a material precedence and averaging scheme is applied to the faces of adjoining cells. In the final meshing step, the faces are broken down into edges. During this meshing procedure, the properties of the materials throughout the grid are converted into electric and magnetic field coefficients assigned to the edges and faces of the cells, respectively.

3.2 Boundary Conditions

Special boundary conditions are usually applied to the fields on the faces of the computational grid. The default boundary condition in LC is the first order Mur absorbing boundary condition [3]. This boundary condition absorbs outgoing energy, in effect extending the edge of the grid indefinitely. The first order Mur introduces very little computational overhead for this benefit, but is only suitable for a restricted set of problems.

The Berenger perfectly matched layers (PML) boundary condition [4, 5] is also available in LC. It is a much more general absorbing boundary condition, and is useful for models with complex materials and geometries. Although it introduces a large amount of computation to the simulation, the algorithm has two useful features for managing the load. First, it is structured in a very analogous way to the FD-TD algorithm. Thus, most of the same techniques for efficiently updating cells in the main FD-TD also apply to the PML region. Second, the number of layers in the absorbing region is user-defined. The number of layers directly influences the amount of memory and computation required for the boundary condition. It also determines the accuracy (its ability to absorb outgoing energy).

3.3 Parallel Code Development

The objective of this effort is to fully develop an interactive electromagnetic (EM) design and analysis tool (LC) that will take complete advantage of the coming class of scalable parallel computers. The efficient implementation of this tool on scalable parallel computers will allow for the examination of problems that are currently too computationally large or complex, conceivably making inroads into some of the grand challenges facing the electromagnetic engineering community.

As part of a contract with NASA through the EMCC (Electromagnetics Code Consortium), Cray developed a parallel FD-TD application program for computing RCS (radar cross section). Implemented using PVM (Parallel Virtual Machine) message passing, the resulting software is portable to all HPC hardware. Code validation and testing proved that the software achieved over 50% of the theoretical peak performance, while scaling linearly up to hundreds of processors [6].

We have leveraged this experience in developing our scalable parallel implementation of LC. The standard version of LC uses implicit parallelism (via compiler directives) to achieve parallel speedup on the Cray shared-memory vector machines, and on smale-scale paralle workstations, such as the Octane and Origin 200. An explicitly parallel implementation of LC, implemented using the parallel Message Passing Interface (MPI) [7], is in progress. This explicitly parallel version currently does not allow for the full interactive GUI present in the standard version of LC, but will correctly process and simulate all standard structures and materials present in the GUI version of LC.

The parallel bookkeeping and passing of data between processors is done using the parallel Message Passing Interface (MPI) [7]. MPI was chosen because it provides good performance combined with portability, allowing the parallel version of LC to run on a wide variety of architectures just as the original serial version of LC did. MPI also has built in functionality to configure the parallel machine as a Cartesian array (i.e., as an $N \times M \times P$ array of processors, where N,M,and P are user selected), in whatever manner is appropriate for the current hardware. This is achieved using a "virtual Cartesian array," created by the `MPI_CREATE_CART` function.

Much of the information about the simulation, such as the grid size and shape, is not known explicitly but deduced at run time. The user can specify the grid size and shape, as well as the structure to analyze at run-time, just as the program is starting execution. If instead the simulation parameters (material values needed, waveform frequency and shape, etc.) were found at compile time and not run time, then every time a parameter changed the user would need to enter all these parameters, recompile the program, and finally run the simulation. By delaying these choices until run time the user is spared a good deal of time and overhead normally involved in the programming/compiling phase.

However, the same run time properties that make design easy for the user make implementation hard for the programmer. If everything had been specified at compile time, such issues as dividing up the problem amongst multiple processors, and inter-processor communication, would have been examined and solved at compile time. However, because the program does not have enough information at compile time, it must wait until run time. As it is running the program must properly redistribute the problem over all of its processors, set up communication channels to transfer data between processors, and funnel the analysis results into various (one to many) output data files. It is this process of data decomposition and inter-process communication that presents the most implementation problems.

The internal data layout of LC records the simulation structure as a list of "blocks" which can represent a material, data probe, source excitation, or several other objects. The overall structure being studied is manipulated by adding, examining, and destroying these blocks. Just before a simulation is run, the representation is frozen and converted into an FDTD mesh, along with some extra data representing the sources, probes, and other information not directly stored in the FDTD mesh. This conversion process—the transforming of a block list into an FDTD mesh, called "meshing" in the program—is where most of the actual programming problems arise for the parallel implementation.

Once the problem is partitioned onto the parallel processors, meshing can proceed just as it did in the serial version. However, one cannot use the original block list on each processor, because then each processor would be meshing and simulating the entire grid. Therefore, each processor obtains a private copy of the block list and modifies that copy to contain only relevant blocks (figure 1). Each processor is responsible for computing a certain region of the total grid; any blocks outside that region are discarded and the remaining blocks are clipped so that they are totally contained in the processor's computational region.

After the block list on each processor has been modified, it is processed normally by using the original serial meshing algorithm. Since the block list only specifies blocks within the processor's computational region, the meshing algorithm will only allocate and mesh that region. The process occurs on all the processors in parallel, with each transforming the block list into a local grid. Proper manipulation of the block list before this step insures that no two processors try to allocate and simulate the same region,

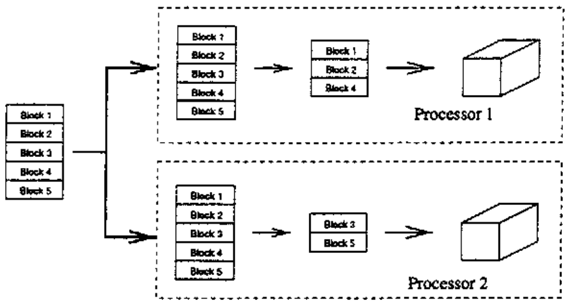

Figure 1: Each processor obtains and clips its own copy of the block list

or overlapping regions.

The process of discarding and clipping elements in the block list will automatically divide up material blocks correctly, so the overall structure under study will be properly partitioned and meshed on the parallel array of processors. However, non-material elements will not function correctly once they have been split onto multiple processors. The serial code will process a given source excitation or data probe block as if it existed only on that processor; if a data probe block is split across two processors, for instance, then each processor will collect probe data for its "local" probe. Some mechanism must exist to gather up the data from all the local probes into the larger probes in the original block list.

Similarly, the source excitation blocks must be configured to spread their voltage or current amplitude across multiple processors. These source blocks are modified to generate a smaller current or voltage, based upon their new size; for instance, if voltage excitation is split exactly in half across two processors, then each of these two new blocks will now generate half the original voltage. Voltage excitations split in other ways will generate voltage amplitudes in proportion to their new size.

Data probe blocks gather their data locally on each processor, and must somehow combine the data into a final, single value. The parallel implementation does this by collecting probe information data from every processor. This probe information lists what probes exist on what processors. If an element such as a data probe spans multiple processors, the element is divided into pieces: one piece of the probe goes onto each processor. During the simulation the pieces combine their data so that they appear and generate output as if they were one single large probe (figure 2. This data is then written out to disk storage or visualized for the user, as it was in the original serial LC implementation.

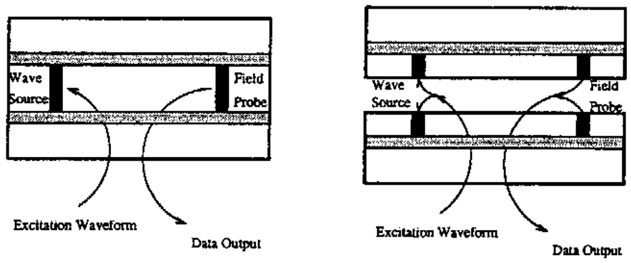

Figure 2: Splitting of arbitrary elements (wave sources and probes) across processor boundaries. Compare a simulation running on one processor (left) and distributed on two processors (right). Arrows show data flow between the processors running the simulation and external stimulus/data storage

3.4 Optimization

Many techniques to speed the FD-TD time-stepping process have been tried in LC. During each half timestep, every field update is independent of the others. Thus substantial speedups can be achived by overlapping the field updates. Vector computers are effective when the grid dimensions are large, because the field updates form a long vector update, hiding memory access time. If the fields are updated in a pattern which results in consecutive memory addresses being used, optimum memory access patterns can be achieved. In Fortran, this means that the inner-most loop of the grid update varies the left-most array index. Cache-based computers with long cache lines, like the IBM RS/6000, this pattern effectively uses the computer's ability to overlap memory access with computation.

Efficient use of a computer requires that the computational requirements are balanced with the memory bandwidth requirements. The large memory bandwidth requirement of FD-TD limits its performance. In a simple FD-TD field update, seven operands are fetched, and one result stored, with only six operations performed. Many computers have the ability to combine a multiply-add operation of the form $a * b + c$ into a single operation. This ability reduces the effective number of operations to four; thus, two values must move for every operation performed.

The number of operands which are read from memory can be reduced while updating a homogeneous region of the grid. This optimization decreases the memory bandwidth requirement relative to the number of operations. In this case, two operand fetches from main memory can be eliminated, reducing the total to five per update. LC automatically detects

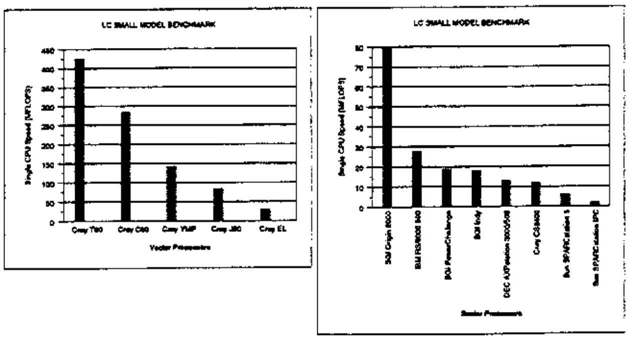

Figure 3: LC electromagnetic simulation performance

global conditions of this type and employs an optimized update.

Since the field updates are independent, parallel FD-TD is straightforward. The grid can be divided into as many regions as necessary, with each region assigned to a processor. In a shared memory computer, the only limiting factor is the size of the regions. If the regions become too small, the processors have insufficient work available between synchronization points. In a distributed memory computer, an additional constraint of communication overhead is present; the field values in the adjacent region are required when updating fields on the edge of a region. Again, when the regions are large, this communication is a small percentage of the overall run time. A hybrid computer architecture, such as the Cray Origin 2000, can be programmed efficiently as a shared memory computer as long as the grid regions are much larger than the hardware page size; most of the memory will reside on the local compute node, with inter-node communication occurring at the region and page boundaries.

3.5 Performance

To obtain some insight into the performance characteristics of the FD-TD solver in LC, a benchmark problem was created. It was designed to be small enough to be practical to run on a desktop computer. This benchmark structure is a microstrip running over a rectangular hole in a groundplane; the size of the simulation is approximately 1.3 Megabytes on architectures using a 32-bit floating point representation. The benchmark was run on as many computers as available, and the results are shown in figure 3. For the multi-CPU computers, the speeds shown are for a single CPU.

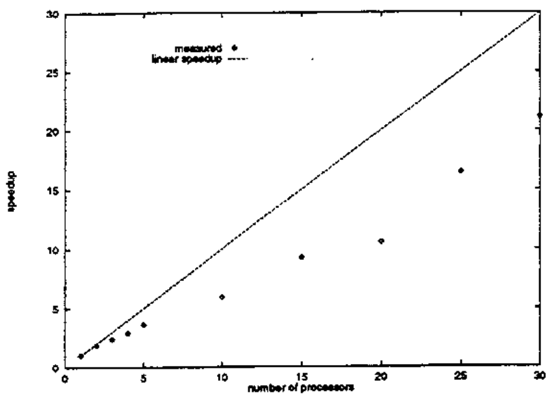

Figure 4: Comparison of ideal (linear) speedup to measure speedup. Measurements were made on a 127 processor Cray Origin 2000 computer.

Early tests using the MPI explicitly parallel form of LC on a small (8 processor) Origin 2000 show a linear to superlinear speedup on a small microstrip simulation. Larger applications do not achieve superlinear performance, but show better scalability than the smaller applications (figure 4). The break point at 20–25 processors is the point where each processor can fit all of its data in local cache, explaining the sudden slope increase in the curve at 25+ processors.

4 Modeling and Parameter Extraction of 3-D Structures

In the current work, a priority was given to three-dimensional modeling accuracy. The current techniques employ a simple method that uses the characteristic definitions of inductance to quantitatively determine the true inductive response directly from the field data. Full-wave solutions to Maxwell's equations are used within the FD-TD engine to accurately determine the true return-path distributions. The user controls the tradeoff between accuracy and computational-resource usage.

4.1 Modeling Signal Path Inductance

Direct calculation of the equivalent-circuit inductance (self or mutual) of any three-dimensional path is obtained with a simple application of first-principles of electromagnetics [8].

Inductance is defined as the ratio of the magnetic flux ϕ through a surface S that links a current I flowing through the surface. The current flowing through the surface S is found by integrating the magnetic field $\vec{H}$ on the closed contour around the surface.

$$\mathcal{L} = \frac{\phi}{I} = \frac{\int_S \vec{B} \cdot ds}{\oint_C \vec{H} \cdot dl} = \frac{\int_S \mu\vec{H} \cdot ds}{\oint_C \vec{H} \cdot dl} \tag{1}$$

In the FD-TD grid, these integrals translate into discrete sums of field values in the FD-TD grid, approximating the continuous field values given in equation (1).

$$\mathcal{L} = \frac{\mu \sum \sum_S \vec{H} \cdot \Delta^2}{\sum_C \vec{H} \cdot \Delta} \tag{2}$$

Where $\Delta = \Delta x = \Delta y = \Delta z$. If the measurements of magnetic flux ϕ and current I are made on a finite length l of transmission line, the inductance per unit length $\mathcal{L}'$ is found from the inductance $\mathcal{L}$ by

$$\mathcal{L}' = \frac{\mathcal{L}}{l} \tag{3}$$

The capacitance is defined as the ratio of the charge in a volume enclosed by a surface to the electromagnetic potential of the surface. The charge in a volume is found by integrating the electric field on the surface enclosing that region, and the potential is found using a line integral of the electric field:

$$\mathcal{C} = \frac{Q}{V} = \frac{\oint_S \vec{D} \cdot ds}{\int_C \vec{E} \cdot dl} = \frac{\oint_S \epsilon\vec{E} \cdot ds}{\int_C \vec{E} \cdot dl} \tag{4}$$

In the FD-TD simulation, these integrals translate into discrete sums of fields values taken from the FD-TD grid:

$$\mathcal{C} = \frac{\sum \sum_S \epsilon\vec{E} \cdot \Delta \cdot \Delta}{\sum_C \vec{E} \cdot \Delta} \tag{5}$$

For example, given a two conductor transmission line as shown in Figure 5, the flux integral would be performed on the surface S with the direction of the surface vector dS as shown. The current integral would be performed along the contour C.

With this method, the challenge of modeling three-dimensional inductance simply reduces to the user properly defining the closed surface over which the flux integration is performed. Call this closed surface the flux net S. The incremental inductance for the segment of trace 1 from A to B is calculated using the flux net defined by the plane ABCD. The return current flows from C to D. The upper and lower edges of this flux net are defined by the edges of the conductors that carry the signal and return-path currents.

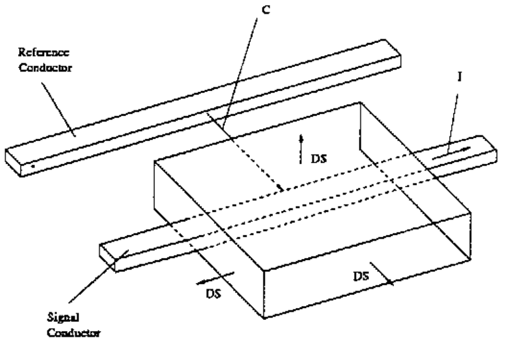

Figure 5: Inductance calculated from flux through S and current on the transmission line

Mutual inductance between two traces, 1 and 2, is modeled by calculating the total amount of flux caught in the flux net defined by ABCD when the signal current I_2 is sent down trace 2. The above illustrative example was simple and two-dimensional in nature. Consider how this simple method is used to quantify the dramatic increase in effective trace inductance when a slot is cut in the ground plane. Now the return path current is diverted around the slot. This new return path converts the simple planer flux net in figure 5 to a hockey net configuration. An example simulation of the above structures uncovers a tripling of effective trace inductance due to the slot in the ground plane.

4.2 Inductance Modeling of three-dimensional Power-Distribution Structures

Determining the equivalent lumped inductance of a three-dimensional structure is of critical importance to electrical engineers involved in package design. Physical and electrical design restrictions can be met and optimal performance can be achieved if this inductance is known before the prototyping stage. Signal-path inductance can be modeled with $L = \frac{\phi}{I}$. In this approach, knowledge of the ground current I return path is required to extract the flux ϕ. This is a fairly straightforward extraction for simple structures, such as a coaxial cable. For more complex models this return path current, and therefore the inductance, is not as easily determined. One such example of a structure without a simple equation for inductance is a power distribution system composed of numerous meshed ground/power planes with connecting vias. This inductance method utilizes a system-level approach involving the analysis of the voltage and current waveforms associated with the structure's inductive element.

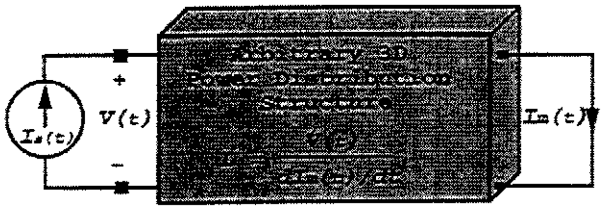

Figure 6: Power distribution structure.

This novel "black-box" method of solving for distributed structure inductance is visualized in the block diagram of (figure 6). In this configuration, an input signal is excited and the voltage (V) across the input is measured. The signal path is shorted out at the output, where the current (I) is measured. With these two sets of data available, the inductance of the system can now be calculated using equation (6):

$$L = \frac{V}{\frac{\delta I}{\delta t}} \tag{6}$$

This approach does not depend on the interior composition of the structure itself, but rather on the ratio of the voltage across the input to the output current time derivative. The voltage-to-current relationship (shown in figure 7) develops as a direct result of the structure's inductive element.

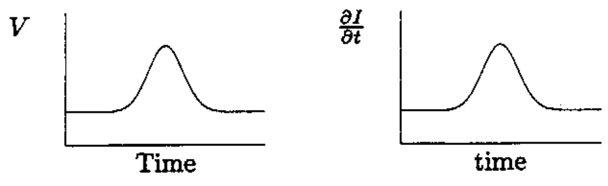

Figure 7: Inductance voltage-to-current relationship; Gaussian pulse input signal.

4.2.1 Meshed PCB Model

A significant problem in modeling and designing high speed digital applications has been understanding how to extract system-level parameters from complex distributed systems. One such complicated structure is shown in figure 8. This structure is a meshed pcb system of three power planes with current sources at the edges, five ground planes, and several vias connecting the power and ground planes. The model

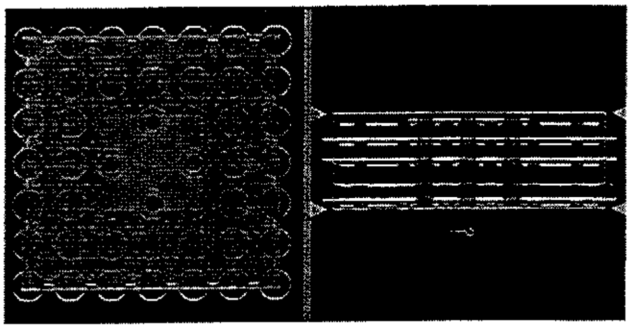

Figure 8: Meshed pcb structure modeled in FD-TD interface

is based on an actual power distribution structure. When the planes are meshed, approximately 50% of the metal is removed. The input voltage waveform is measured between a power plane and a ground plane. A via short to a ground plane on top of the structure provides the current response which is due to the structure's inductive element. Until now, the inductance calculation for this type of structure has been done on a piecewise scale; that is, by examining the effects of the power/ground planes on the vias and vice versa.

However, by applying this new system-level method to the system, it is possible to represent the entire structure as a lumped inductor with one straightforward calculation. After running the FD-TD simulation and extracting the required voltage and current measurements, the inductance of a 3mm × 3mm subsection of a PCB was found in the numerical model to be 83.8pH.

It is clear that this approach can be useful in determining the optimal physical parameters of a system while staying within electrically-dictated inductance thresholds that often accompany high speed digital applications. This method can be integral in the development of certain design guidelines for structures similar to the one in figure 8.

5 Conclusion

This work has direct application for analysis and design of advanced electronic components such as multichip modules (MCM) and mixed signal modules. The capabilities of LC address requirements that are currently being pursued in both the defense and commercial sectors. It is directly applicable to existing efforts in design and analysis of advanced packaged modules, low cost electronic packaging, phased array antennas, MCMs, mixed signal modules, and the study of electromigration phenomenology. We eventually envision a tool that will not only be able to address the complex electromagnetic interactions within individual devices and components but will also address coupling of energy into electronic devices from external sources (for example noise or jamming signals).

We are planning to investigate the use of grid partitioning to employ specialized field updates to reduce the amount of memory and memory bandwidth required during the simulation. Using this technique, the amount of memory required can be reduced by up to two thirds, and the processing speed can be increased by a factor of two.

We plan to add new algorithm and postprocessing features that are requested by users. One algorithm directly relevant to modeling power and signal distribution structures involves taking into account losses due to skin effects. Another one is the modeling of dispersive materials [9, 10, 11].

LC has been the culmination of approximately a decade of electromagnetics research and development at Cray. This experience includes CRADAs with National Labs, government funding of Radar Cross Section (RCS) work, and code development at Northwestern University and the University of Colorado at Boulder.

References

[1] Allen Taflove, *Computational Electrodynamics the Finite-Difference Time-Domain Method*, pp. 98–100,111–134,281–295, Artech House, Boston, 1995.

[2] V.A. Thomas, M.E. Jones, M.J. Piket-May, A. Taflove, and E. Harrigan, "The use of spice lumped circuits as sub-grid models for fd-td analysis," *IEEE Microwave and Guided Wave Letters*, vol. 4, pp. 141–143, 1994.

[3] G. Mur, "Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic field equations," *IEEE Trans. on Electromagnetic Compatibility*, vol. 23, pp. 377–382, 1981.

[4] D.S. Katz, E.T. Thiele, and A. Taflove, "Validation and extension to three dimensions of the berenger pml absorbing boundary condition for fd-td meshes," *IEEE Microwave and Guided Wave Letters*, vol. 4, pp. 268–270, 1994.

[5] J.P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of Computational Physics*, vol. 114, pp. 185–200, 1994.

[6] S. Barnard and A. Taflove, "Application of massively parallel supercomputing to 3D RCS methods and modeling of complex materials," Final report on nasa-ames contract nas213890, Dec. 1994.

[7] "The message passing interface (MPI) standard 1.2 and 2.0," http://www.mpi-forum.org/.

[8] Roger Gravrok, Melinda Piket-May, and Kevin Thomas, "LC: An integrated methodology to model and visualize the complex electrodynamics of 3D structures," in *Electrical Performance of Electronic Packaging*. IEEE MTT Society, Oct. 1995.

[9] X. Zhang, J. Fang, K.K. Mei, and Y. Liu, "Calculation of the dispersive characteristics of microstrips by the time-domain finite-difference method," *IEEE Trans. on Microwave Theory Tech.*, vol. 36, pp. 263–267, 1988.

[10] T. Kashiwa and I. Fukai, "A treatment by FDTD method of dispersive characteristics associated with electronic polarization," *Microwave and Optics Technologies Letters*, vol. 3, pp. 203–205, 1990.

[11] R. Luebbers, F. Hunsberger, K. Kunz, R. Standler, and M. Schneider, "A frequency-dependent finite-difference time-domain formulation for dispersive materials," *IEEE Trans. on Electromagn. Compat.*, vol. 32, pp. 222–229, 1990.